**Week 2**

**JUnit Testing Exercises**

**Exercise 1: Setting Up Junit**

**Scenario:** You need to set up JUnit in your Java project to start writing unit tests.

**Step1** : Create a new Java project in your IDE(Eclipse)

**Step2** : Add JUnit dependency to your project.

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.junitsetup</groupId>

<artifactId>practice</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

**Step 3:**

**GreetingService.java**

package practice;

public class GreetingService {

public String greet(String name) {

if (name == null || name.isEmpty()) {

return "Hello, Guest!";

}

return "Hello, " + name + "!";

}

}

**GreetingServiceTest.java**

package practice;

import static org.junit.Assert.*assertEquals*;

import org.junit.Test;

public class GreetingServiceTest {

@Test

public void testGreetWithName() {

GreetingService service = new GreetingService();

String result = service.greet("Madhumala");

*assertEquals*("Hello, Madhumala!", result);

}

@Test

public void testGreetWithEmptyString() {

GreetingService service = new GreetingService();

String result = service.greet("");

*assertEquals*("Hello, Guest!", result);

}

@Test

public void testGreetWithNull() {

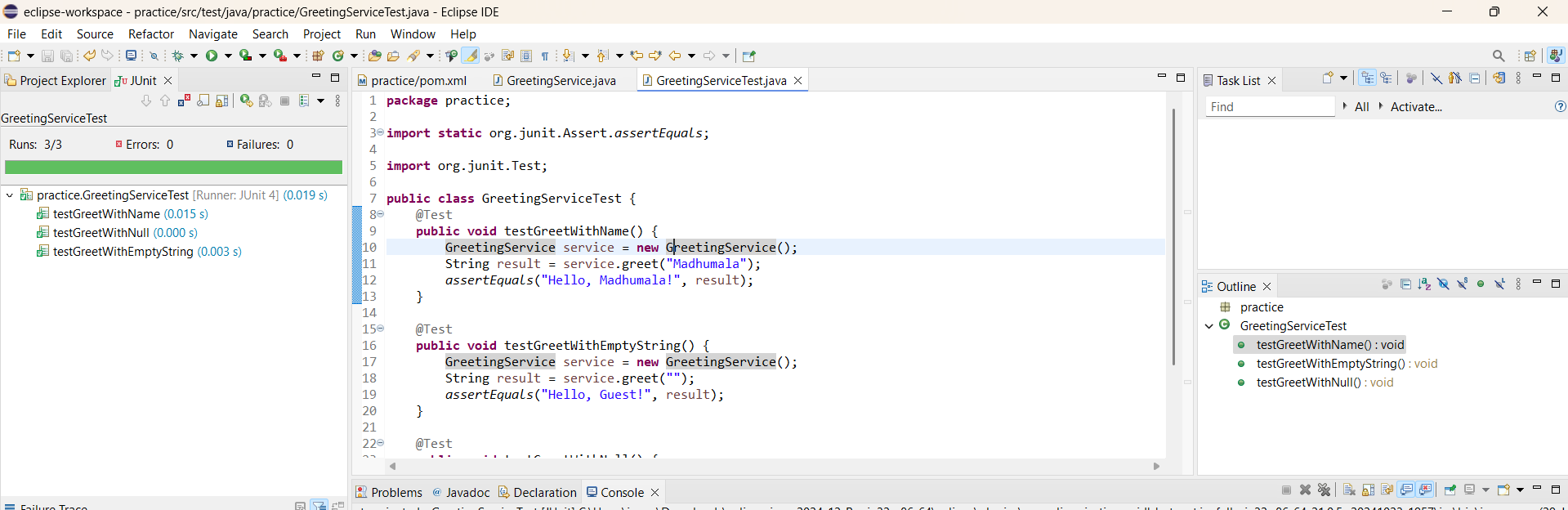
GreetingService service = new GreetingService();

String result = service.greet(null);

*assertEquals*("Hello, Guest!", result);

}

}

****

**Exercise 3: Assertions in Junit**

**Scenario:** You need to use different assertions in JUnit to validate your test results.

**AssertionsTest.Java**

**package** practice;

**import** org.junit.Test;

**import** **static** org.junit.Assert.\*;

**public** **class** AssertionsTest {

@Test

**public** **void** testVariousAssertions() {

*assertEquals*("Sum should be 10", 10, 7 + 3);

*assertTrue*("This should be true", "OpenAI".startsWith("O"));

*assertFalse*("Should be false", 100 < 50);

String name = **null**;

*assertNull*("Name should be null", name);

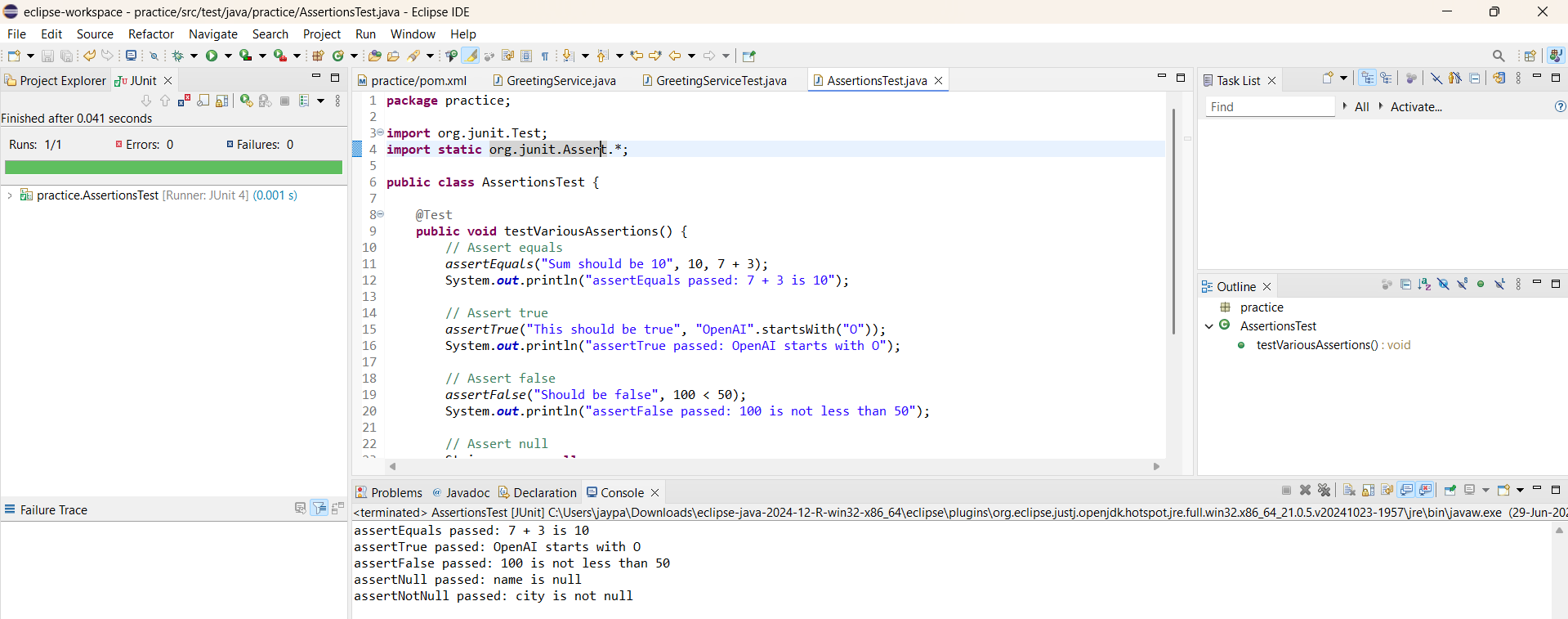
String city = "Hyderabad";

*assertNotNull*("City should not be null", city);

}

}

**Output:**



**Exercise 4: Arrange-Act-Assert (AAA) Pattern, Test Fixtures, Setup and Teardown Methods in Junit**

**Scenario:** You need to organize your tests using the Arrange-Act-Assert (AAA) pattern and use setup and teardown methods.

**ServiceTest.java**

package practice;

public class ServiceTest {

public String greet(String name) {

return (name == null || name.isEmpty()) ? "Hello, Guest" : "Hello, " + name;

}

}

**ExampleServiceTest.java**

package practice;

import org.junit.Before;

import org.junit.After;

import org.junit.Test;

import static org.junit.Assert.\*;

public class ExampleServiceTest {

private ServiceTest service;

@Before

public void setUp() {

service = new ServiceTest();

}

@After

public void tearDown() {

service = null;

}

@Test

public void testGreetWithName() {

String input = "John";

String result = service.greet(input);

*assertEquals*("Hello, John", result);

System.out.println("testGreetWithEmpty passed with result: " + result);

}

@Test

public void testGreetWithEmpty() {

String input = "";

String result = service.greet(input);

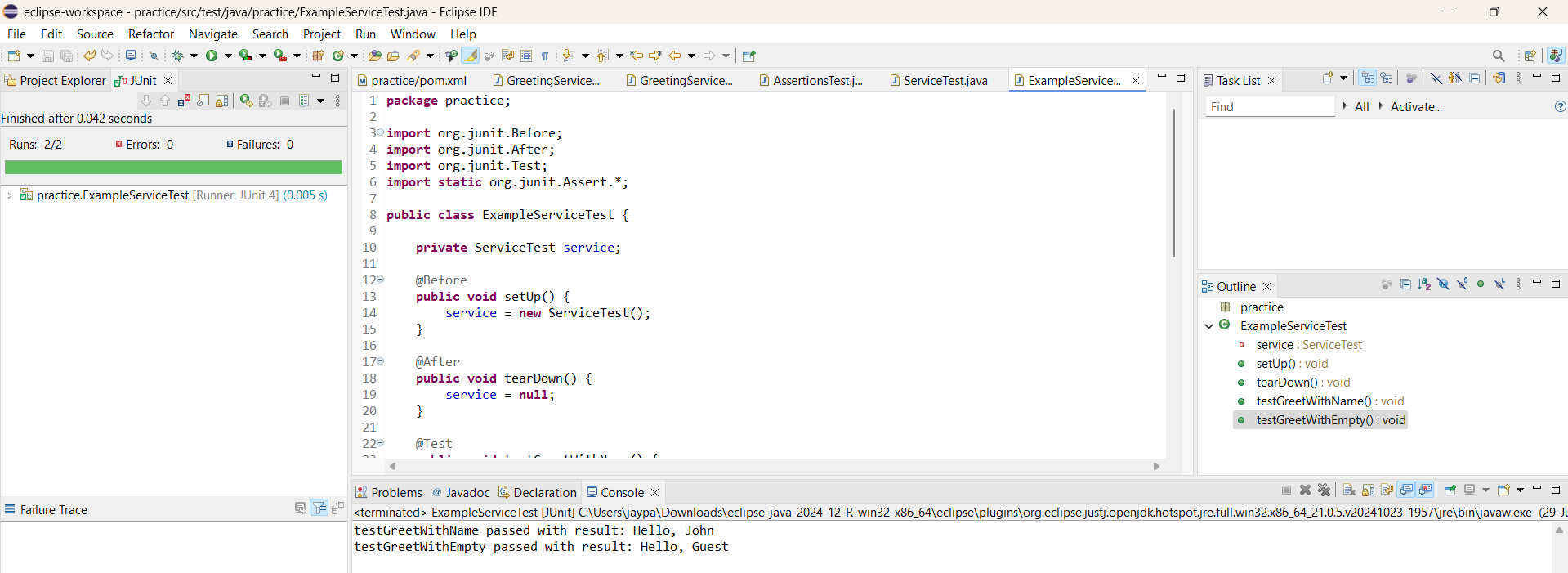
*assertEquals*("Hello, Guest", result);

System.out.println("testGreetWithEmpty passed with result: " + result);

}

}

**Output:**



**Mockito Exercises**

**Exercise 1: Mocking and Stubbing**

**Scenario:** You need to test a service that depends on an external API. Use Mockito to mock the external API and stub its methods.

**Pom.xml:**

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter</artifactId>

<version>5.9.1</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>4.8.1</version>

<scope>test</scope>

</dependency>

**ExternalAPI.java:**

package practice;

public interface ExternalApi {

String getData();

}

**MyService.java:**

package practice;

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

**MyServiceTest.java**

package practice;

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.\*;

import static org.mockito.Mockito.\*;

public class MyServiceTest {

@Test

public void testFetchDataReturnsMockedValue() {

ExternalApi mockApi = *mock*(ExternalApi.class);

*when*(mockApi.getData()).thenReturn("Mocked Response");

MyService service = new MyService(mockApi);

String result = service.fetchData();

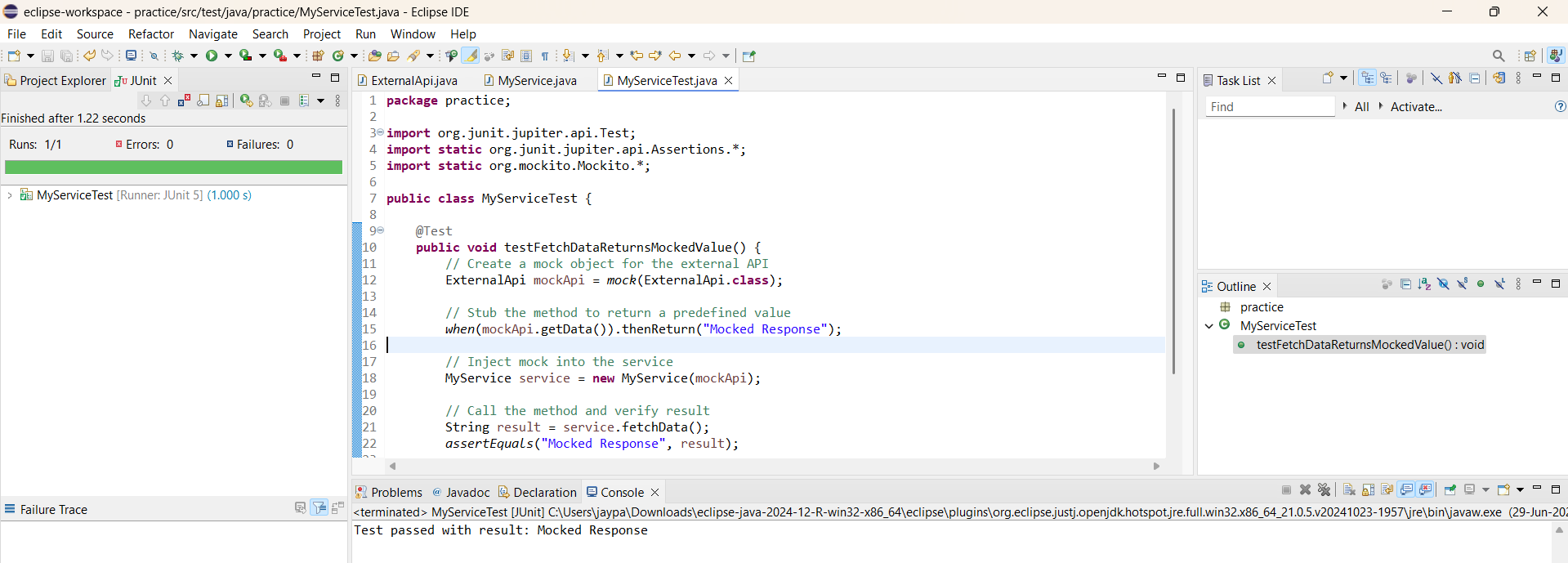
*assertEquals*("Mocked Response", result);

System.*out*.println("Test passed with result: " + result);

}

}

**Output:**



**Exercise 2: Verifying Interactions**

**Scenario:** You need to ensure that a method is called with specific arguments.

**ExternalAPI.java:**

package practice;

public interface ExternalApi {

String getData();

}

**MyService.java:**

package practice;

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

**MyServiceTest1.java:**

package practice;

import org.junit.jupiter.api.Test;

import static org.mockito.Mockito.\*;

public class MyServiceTest1 {

@Test

public void testVerifyInteraction() {

ExternalApi mockApi = *mock*(ExternalApi.class);

MyService service = new MyService(mockApi);

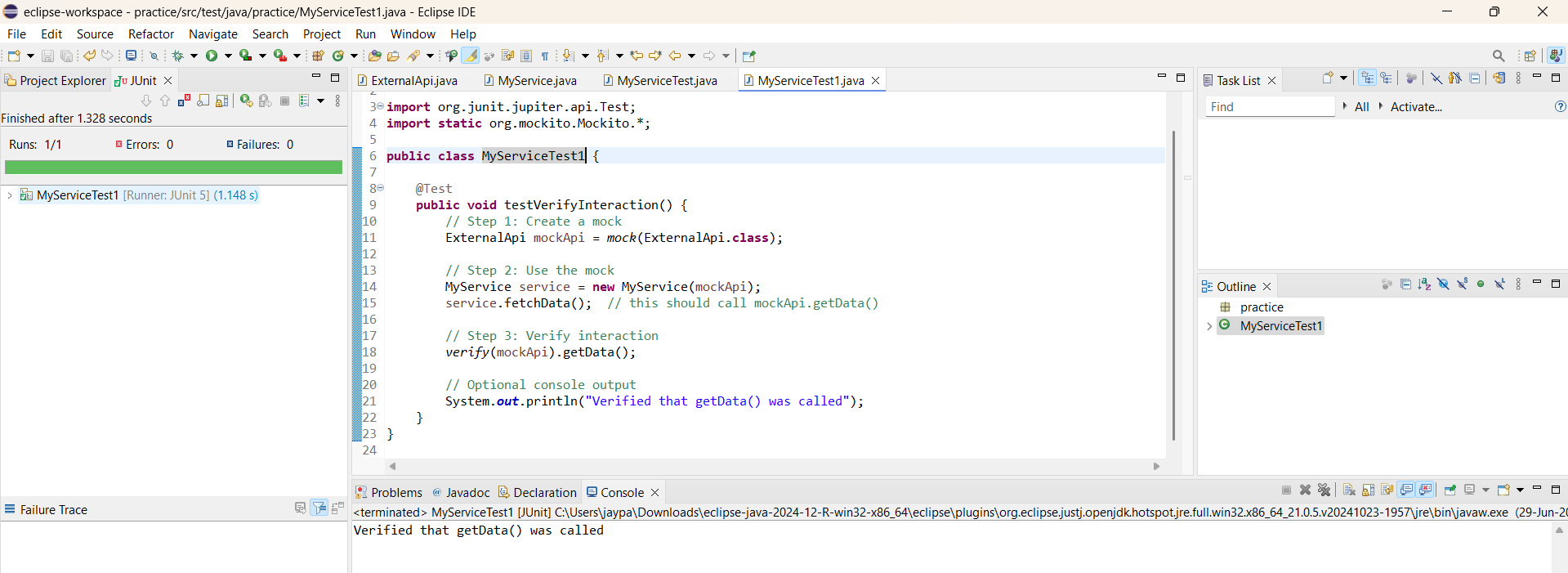
service.fetchData();

*verify*(mockApi).getData();

System.*out*.println("Verified that getData() was called");

}}

**Output:**

****

**Logging using SLF4J**

**Exercise 1: Logging Error Messages and Warning Levels**

**Task:** Write a Java application that demonstrates logging error messages and warning levels using SLF4J.

**pom.xml :**

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>1.7.30</version>

</dependency>

<dependency>

<groupId>ch.qos.logback</groupId>

<artifactId>logback-classic</artifactId>

<version>1.2.3</version>

</dependency>

**LoggingDemo.java**

package practice;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

public class LoggingDemo {

private static final Logger logger = LoggerFactory.getLogger(LoggingDemo.class);

public static void main(String[] args) {

logger.error("Error: Unable to connect to the server");

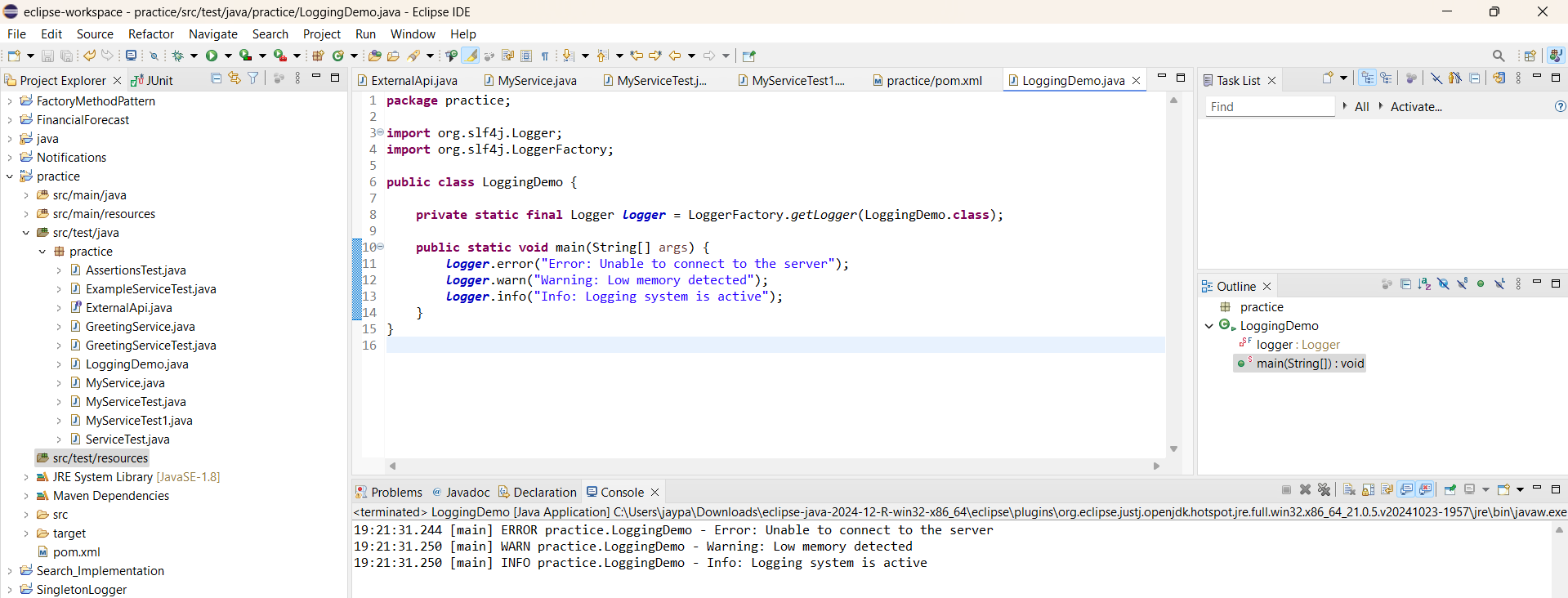
logger.warn("Warning: Low memory detected");

logger.info("Info: Logging system is active");

}

}

**Output:**

****